41872/FLC/I281

METHOD AND APPARATUS FOR COMPUTER NETWORK ANALYSIS

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application No. 60/262,112 filed on January 16, 2001 which is hereby incorporated by reference as if set forth in full herein.

BACKGROUND OF THE INVENTION

This invention relates generally to the field of computer network analysis and more specifically to the analysis of computer networks using a hybrid method employing analytical and discrete event simulation methodologies.

Global Frame Relay (GFR) services provide a class of service functionality between any two sites in the world that communicate across a computer network. This class of service functionality, called Customized Networking Options (CNOs), provides the ability to segregate network traffic across different prioritized GFR permanent virtual circuits (PVCs) based on various protocol and software application criteria.

An advantage of segregating software application traffic across different priority GFR CNOs is that during times of network congestion higher priority traffic receives preferential treatment over lower priority traffic. Software applications using a network comprising GFR CNOs exhibit different performance characteristics. In order to design a network correctly, it is advantageous to understand the nuances of a software application that will run over the network.

Therefore, a need exists for a network analysis system to monitor and analyze the actual network demands of a running software application within a real-world networking environment. The present invention meets such need.

SUMMARY OF THE INVENTION

In one aspect of the invention, a method is provided for determining a computer network's performance during operation of a software application using the computer network. The method includes recording network traffic data while the software application is using the computer network. A latency sensitivity metric is generated from the network traffic data. A bandwidth sensitivity metric is generated from the network traffic data as well. The latency sensitivity metric and the bandwidth sensitivity metric are then compared to determine the computer network's performance during operation of the software application.

In another aspect of the invention, the method for determining a computer network's performance during operation of a software application using the computer network includes calculating the latency sensitivity metric by generating from the network traffic data a plurality of computer network response times for a plurality of software application use scenarios at a constant computer network bandwidth value and generating the latency sensitivity metric from the plurality of computer network response times. In one embodiment of the method, the latency sensitivity metric is generated by calculating the standard deviation of the plurality of network response times. In another embodiment of the method, the latency sensitivity metric is generated by calculating the slope of a line defined by plotting the plurality of computer network response times versus the plurality of software application use scenarios.

In another aspect of the invention, the method for determining a computer network's performance during operation of a software application using the computer network includes calculating the bandwidth sensitivity metric by generating from the network traffic data a plurality of computer network response times for a plurality of computer network bandwidth values for

a software application use scenario and generating the bandwidth sensitivity metric from the plurality of computer network response times. In one embodiment, the bandwidth sensitivity metric is generated by calculating the standard deviation of the plurality of network response times. In another embodiment, the bandwidth sensitivity metric is generated by calculating the slope of a line defined by plotting the plurality of computer network response times versus the plurality of computer network bandwidth values.

In another aspect of the invention, the network traffic data is used to generate a return on investment data table for use in generating a return on investment metric and a computer network simulation table for use in simulating a second computer network.

In another aspect of the invention, a data processing system is adapted to determine a computer network's performance during operation of a software application using the computer network. The data processing system includes a processor and a memory operably coupled to the processor. The memory has program instructions stored therein and the processor is operable to execute the program instructions. The program instructions include: receiving network traffic data recorded while the software application is using the computer network; generating from the network traffic data a latency sensitivity metric; generating from the network traffic data a bandwidth sensitivity metric; and comparing the latency sensitivity metric and the bandwidth sensitivity metric to determine the computer network's performance during operation of the software application.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a network diagram of an exemplary test environment network for collecting network traffic data;

FIG. 2 is a process flow diagram of one embodiment of a process for collecting network traffic data according to the present invention;

FIG. 3 is a sequence diagram depicting an embodiment of a network traffic data collection process according to the present invention;

FIG. 4 is a data flow diagram depicting the data flow in one embodiment of the network analysis program;

FIG. 5 depicts an embodiment of a sensitivity data table extracted from the network traffic data stored by the network analyzer during an actual test environment run;

FIG. 6 is a plot of the duration of a software application request and server response transaction versus test scenario taken from the first row of the sensitivity data table of FIG. 5;

FIG. 7 is a plot of the duration of a software application request and server response transaction versus effective bandwidth of the test environment network taken from the first column of the sensitivity data table of FIG. 5;

FIG. 8 is a process flow diagram of one embodiment of the process of determining if the software application using the test environment network is most sensitive to the latency or the bandwidth of the test environment network;

FIG. 9 is a hardware architecture diagram of a general purpose computer suitable for use network analysis program host; and

APPENDIX A is an exemplary embodiment of an analysis report generated from the data collected from an analysis.

DETAILED DESCRIPTION OF THE INVENTION

The present invention is a system used to collect and analyze network traffic data collected from a test environment network wherein the network traffic within the test environment

network includes network traffic generated by application transactions. The resultant analysis is used to design data networks based on a software application centric approach.

A task of network design is to accurately predict how software applications will perform under certain conditions. Network modeling can be performed using several different methodologies. One network modeling methodology is to use mathematical algorithms to estimate link (or virtual circuit) utilization and network latency. However, protocol effects are difficult to capture. Important protocol aspects that are extremely difficult to represent in a mathematical network model include data segmentation, congestion control, re-transmissions, load balancing across multiple routes, and sophisticated algorithms employed within different protocol layers such as selective acknowledgments in Transmission Control Protocol(TCP) or weighted fair queuing in Internet Protocol (IP).

Another network modeling methodology is to use discrete event simulation methods. By either manually building the unique characteristics of a network and its various components or drawing upon a library of predefined components, explicit network traffic is generated creating an accurate baseline of network traffic data. Once this baseline is created, multiple network design scenarios can be simulated in order to determine a multitude of network and application metrics such as application response time, link utilization, and throughput. These scenarios can include an increase in the user population over time or the addition of new software applications. The advantage of using discrete event simulation methods is that the model can accurately reflect the uniqueness and nuances of a specific software application and/or network.

The present invention is a modeling tool that uses a hybrid approach combining analytical and discrete event simulation methods.

FIG. 1 depicts one embodiment of a test environment for collection of network traffic data. The test environment is used to collect network traffic data during actual usage of software applications running on a computer network. The test environment is configured to match the logical and physical configuration of a portion of an actual network as the actual network is used in a real-world implementation. In one exemplary embodiment according to the present invention, a server 100 is operably coupled to a client 120 via a computer network comprising a networking device such as a network switch 110. The network switch is operably coupled to a network analyzer 130 that collects network traffic data and stores the network traffic data on a permanent data storage device 140 for later use.

In operation, the client hosts a software application that requests and receives data from the server. The application is invoked on the client and a suite of test operations are run using the software application. The test operations simulate the actual demand placed on the software application and network by a user. The network analyzer records statistics about the data traffic within the network between the client and server and stores data describing the network traffic in a network traffic database or file of network traffic traces. The network traffic data is used as the input for a combination of analysis and reporting tools.

In other embodiments of the present invention, the number of software applications, clients, servers, network devices, and network analyzers are varied within the test environment to recreate the elements of an actual computer network. Furthermore, the suite of tests run using an application is varied to match actual usage scenarios.

FIG. 2 is a process flow diagram of one embodiment of a process for collecting network traffic data according to the present invention. The test environment is configured during

step 200 and a suite of test scenarios are run during step 210. The network analyzer collects information about application generated transactions and network traffic data and stores the network traffic data 220. A determination is made of whether or not to continue collecting network traffic data at step 230. If the test is finished, the collected network traffic data is analyzed at step 240 and reports 250 describing the test environment network are created. If more network traffic data is to be collected, the test environment network is modified at step 250 and a new suite of test scenarios are run at step 210.

In one embodiment of the present invention, the suite of test scenarios is designed to collect additional network traffic data under increasing load demands on an existing network test environment configuration.

In one embodiment of the present invention, the test environment is modified in steps with increasing amounts of effective bandwidth made available to a software application running within the test environment.

FIG. 3 is a sequence diagram depicting an embodiment of a network traffic data collection process according to the present invention. A client 120 sends a data request 300 to a server 100 via a network device such as switch 110. A network analyzer collects statistics about the data request 302 from the switch and stores the data request statistics 304 in a network traffic data store 140. The switch forwards the data request 306 to the server and the server sends response data 308 to the client via the switch. The network analyzer collects statistics 310 about the data being sent to the client from the server and stores the data statistics 312 in the network traffic data store. The switch forwards the response data 314 to the client. The request and response cycle is repeated continuously and data statistics are stored for each request and response. The stored statistics

comprise a data trace describing the network traffic during the test scenarios.

In one embodiment of the present invention, the data statistics include a time stamp for each request and data statistic collected and stored by the network analyzer.

In one embodiment of the present invention, the data statistics include the size in bytes of each request, the size in bytes of the overhead associated with each request, the total size in bytes of the response data, the size in bytes of the overhead associated with each response, and the duration in seconds of the request/response interaction.

In one embodiment of the present invention, a Sniffer Pro LAN seven layer network analyzer as supplied by Sniffer Technologies of Santa Clara, California is used to collect network traffic data.

FIG. 4 is a data flow diagram depicting the data flow in one embodiment of the network analysis program. Data is collected from the previously described test environment network in the previously described data collection process 400. Portions of the data files 402 pertaining to a single software application's requests and server responses are used by a modeling and simulation analysis tool 416 to make network design decisions based on network modeling and simulation scenarios. An exemplary modeling and simulation analysis tool suitable for use is "IT DecisionGuru" available from OPNET Technologies Inc. of Washington DC, USA.

A database creator 404 is used to create a database 406 from the network traffic data 402 collected by the network analyzer. An exemplary embodiment of a database creation tool suitable for use is Application Expert supplied by Compuware Corporation of Farmington Hills, MI, USA. A proprietary data extractor 420 is used as a front end to the database by other applications and data analysis tools. The data extractor searches the database

for data appropriate for each application or data analysis tool. The appropriate data is then used to create a data file suitable for each application or data analysis tool.

In one embodiment of a data extractor according to the present invention, the data extractor applies a weighting function to the statistics generated from the network traces created during the data collection phase. For example, if the test scenario included use of a client application to enter data into a database using a database server and a client application for generating a report from the database, the use of the client applications will differ. The reporting client application may be used ten times as much as the data entry client application during a given time period. In this case, the data extractor weights the reporting client application's network usage ten times heavier than the data entry client application's network usage.

An exemplary output report created by the data extractor is a profile data report 422 containing a synopsis of the data traces collected during the test phase of the network analysis. The profile data report includes the following information in a tabular format:

Column Name	Description
· Total Bytes	Total bytes transmitted.
· App Turns	Total application turns.
· Bytes/App Turn	Calculation based on Total Bytes / App Turns.
· Client Bytes - Total	Total bytes sent from the client to the server.
· Client Bytes - Payload	Total amount of application data sent from the client to the server.
· Client Bytes - Overhead	Calculation based on the sum of the Client Bytes - Client Payload Bytes / Client Bytes expressed as a percentage.
· Server Bytes - Total	Total bytes sent from the server to the client.

· Server Bytes - Payload	Total amount of application data sent from the server to the client.
· Server Bytes - Overhead	Calculation based on the sum of the Server Bytes - Server Payload Bytes / Server Bytes expressed as a percentage.
· Ratio	Calculation based on either the Client Bytes / Server Bytes OR Server Bytes / Client Bytes depending on the initial values of Client Bytes and Server Bytes.
· Duration	Total amount of elapsed time (in seconds).
· Overall Protocol Overhead	Calculation based on the sum of the (Server + Client Bytes) - (Server + Client Payload Bytes) / (Server + Client Bytes) expressed as a percentage.
· Overall Server to Client Ratio	Calculation based on either the Total Client Bytes / Total Server Bytes OR Total Server Bytes / Total Client Bytes depending on the initial values of Total Client Bytes and Total Server Bytes.
· Average Client Transaction	Geometric mean of Client Total Bytes based on all individual tasks.
· Average Server Transaction	Geometric mean of Server Total Bytes based on all individual tasks.
· RV	Specifies the Report Version

Another exemplary output report created by the data extractor is a custom application data report 424 containing information used in the creation of custom applications within the previously described IT DesignGuru modeling and simulation tool. The custom application data report includes the following information in a tabular format:

Column Name	**Description**
· Duration	Total amount of elapsed time (in seconds).

· Total Bytes	Total payload bytes transmitted.
· Payload - Client	Total amount of application data sent from the client to the server.
· Payload - Server	Total amount of application data sent from the server to the client.
· Frames - Client	Total number of packets sent from the client to the server.
· Frames - Server	Total number of packets sent from the server to the client.
· Packet Size - Client	Average size of payload information that was sent from the client to the server.
· Packet Size - Server	Average size of payload information that was sent from the server to the client.

A sensitivity analyzer 408 is used to determine whether or not the software applications simulated and used in the test environment are most sensitive to network latency or to network bandwidth. The sensitivity tool analyzes the data traces taken from the test system and determines for each application whether or not the application is most sensitive to latencies in data transfers or to the effective bandwidth of the computer network. The results from the sensitivity analysis tool are displayed in a sensitivity data table 410.

FIG. 5 depicts one embodiment of a sensitivity data table extracted from the network traffic data stored by the network analyzer during an actual test environment run. The sensitivity data table contains the network traffic data used to determine if a software application using the test environment network is most sensitive to network latency or to network bandwidth. The sensitivity data table comprises measured durations for a request/response interaction between a software application and a server under different simulated test environment configurations and testing scenarios. The effective bandwidth

of the test environment network is fixed and then a series of simulated test scenarios are run creating varying demands on the test environment network. The fixed effective bandwidth of the test environment network is placed in column 500. In this example, the effective bandwidth starts at 56 Kbps and is increased in steps to 2,048 Kbps. Subsequent columns 502, 503, 506, 508, and 510 contain the duration values for test scenarios comprising the test suite. In this example, the duration values from the first test scenario are in column 502.

Reading the table along a single row of the sensitivity data table indicates the sensitivity of the application to the latency of the network. For example, reading the data along row 504 of the sensitivity data table shows that duration of a request/response transaction increases from 227.66 seconds to 239.91 seconds as the latency of the network increases for a fixed bandwidth of 56 Kbps.

FIG. 6 is a plot of duration versus test scenario taken from the first row of the sensitivity data table of FIG. 5. The duration in seconds is along the Y axis 600 and the scenario number is along the X axis 602. The resultant line 604, has a positive slope because the duration of a software application request and server response interaction increases as the load on the test environment network increases. This indicates that the software application being run on the test environment network is at least partially sensitive to the latency of the test environment network.

Returning to FIG. 5, reading the sensitivity data table along a column indicates the sensitivity of a software application to the effective bandwidth of the test environment network. For example reading down column 502 the duration of a software application request and server response transaction decreases from 227.66 seconds to 206.48 seconds as the effective

bandwidth of the test environment network is increased from 56 Kbps to 2048 Kbps. An example of this is given in FIG. 7.

In FIG. 7, the duration of a software application request and server response transaction is plotted against the effective bandwidth of a test environment network. The duration in seconds of a client/server request and response interaction is along the Y axis 700 and the effective bandwidth of the test environment network is along the X axis 702. The resultant line 704, has a negative slope because the duration of a client/server request and response interaction decreases with increasing bandwidth. This indicates that the application being simulated on the test environment network is at least partially sensitive to the effective bandwidth of the test environment network.

Returning to FIG. 5, the data in the sensitivity table is used to determine whether or not the software application using the test environment network is most sensitive to the latency or the bandwidth of the test environment network. A sensitivity metric is calculated in order to make this determination. The standard deviation of the average value of the duration of a client/server request and response interaction for a column of the sensitivity data table is an indication of how sensitive the software application using the test environment network is to the effective bandwidth of the test environment network. The higher the standard deviation, the greater the sensitivity. For example, the standard deviation of the durations of software application request and server response interactions for column 502 is 6.48 seconds as shown in field 512. Similarly, the standard deviations of the durations of software application request and server response interactions within a row is an indication of how sensitive the software application using the test environment network is to the latency of the test environment network. For example, the standard deviation of the

durations of client/server request and response interactions for row 504 is 5.01 seconds as shown in field 518 in this example.

The combined values of the standard deviations of the durations of software application request and server response interactions along the columns of the sensitivity data table serves as a bandwidth sensitivity metric and the combined values of the standard deviations for the durations of software application request and server response interactions along the rows of the sensitivity data table serves as a bandwidth sensitivity metric. In this example, the average standard deviation for the columns of the sensitivity data table is 6.31 seconds as shown in field 514. This value is used as the bandwidth sensitivity metric for this example. The average standard deviation for the rows of the sensitivity data table is 5.47 seconds as shown in field 520. This value is used as the latency sensitivity metric for this example.

The bandwidth sensitivity metric and the latency sensitivity metric are compared to determine whether the software application using the test environment network is most sensitive to the latency or the bandwidth of the test environment network. In this case, the value of the bandwidth sensitivity metric is 6.31 seconds and the value of the latency sensitivity metric is 5.47 seconds. Therefore, the software application using the test environment network is most sensitive to the bandwidth of the test environment network because the value of the bandwidth sensitivity metric is 6.31 seconds which greater than the value of the latency sensitivity metric of 5.47 seconds. This is shown in the sensitivity data table in the sensitivity result field 522 where the word "BANDWIDTH" is displayed.

In one embodiment of the present invention, the latency and bandwidth sensitivity metrics are calculated as the absolute value of the slope of a best fit line calculated from the

durations of software application request and server response interactions.

FIG. 8 is a process flow diagram of one embodiment of the process of determining if the software application using the test environment network is most sensitive to the latency or the bandwidth of the test environment network. A previously described latency sensitivity metric is calculated 800 using the data from the previously described sensitivity data table. A previously described bandwidth sensitivity metric is calculated 802 using the data from the previously described sensitivity data table. The two values are compared 804. If the latency sensitivity metric is greater than the bandwidth latency sensitivity metric then the application using the test environment network is determined to be most sensitive the latency of the network 808. If the latency sensitivity metric is less than the bandwidth sensitivity metric then the application using the test environment network is determined to be most sensitive the bandwidth of the test environment network 806.

Referring again to FIG. 4, the data extractor creates a Return On Investment (ROI) application report 412 using the trace data collected from the test system. The ROI application report is used as an input into an ROI analysis tool 414 to compare different network configurations from a financial perspective. The ROI analysis tool calculates a financial rate of return on investment for each network configuration in order to help in the decision making process of network service procurement. An exemplary embodiment of a ROI tool is "ROI Solution Builder" available from Infonet Services Corporation of El Segundo, California USA. The ROI data report includes the following information in a tabular format:

Column Name	**Description**
· Duration	Total amount of elapsed time (in

	seconds).
· App Turns	Total application turns.
· Client - Bytes	Total bytes sent from the client to the server.
· Client - Payload	Total amount of application data sent from the client to the server.
· Client - Packets	Total number of packets sent from the client to the server.
· Client - Avg. Payload	Average amount of application data sent from the client to the server.
· Client - OHD	Calculation of the average amount of overhead bytes for a given task that is sent from the client to the server based on the total client bytes - total client payload bytes / total number of client packets.
· Server - Bytes	Total bytes sent from the server to the client.
· Server - Payload	Total amount of application data sent from the server to the client.
· Server - Packets	Total number of packets sent from the server to the client.
· Server - Avg. Payload	Average amount of application data sent from the server to the client.
· Server - OHD	Calculation of the average amount of overhead bytes for a given task that is sent from the server to the client based on the total server bytes - total server payload bytes / total number of server packets.
· Avg. Client OHD	Calculation of the average amount of overhead bytes for the entire application that is sent from the client to the server based on the total client bytes - total client payload bytes / total number of client packets.
· Avg. Server OHD	Calculation of the average amount of overhead bytes for the entire application that is sent from the server to the client based on the total server bytes - total server payload bytes / total number of server packets.

The output from each of the tools is combined into a final report 418. An exemplary final report is included in APPENDIX A.

FIG. 9 is a hardware architecture diagram of a general purpose computer suitable for use network analysis program host. A microprocessor 900, including a Central Processing Unit (CPU) 910, a memory cache 920, and a bus interface 930, is operatively coupled via a system bus 935 to a main memory 940 and a I/O control unit 945. The I/O interface control unit is operatively coupled via a I/O local bus 950 to a disk storage controller 995, and a network controller 980.

The disk storage controller is operatively coupled to a disk storage device 925. Computer program instructions 997 for implementing a network analysis program are stored on the disk storage device until the microprocessor retrieves the computer program instructions and stores them in the main memory. The microprocessor then executes the computer program instructions stored in the main memory to implement the features of a network analysis program.

Although this invention has been described in certain specific embodiments, many additional modifications and variations would be apparent to those skilled in the art. It is therefore to be understood that this invention may be practiced otherwise than as specifically described. Thus, the present embodiments of the invention should be considered in all respects as illustrative and not restrictive, the scope of the invention to be determined by any claims supportable by this application and the claims' equivalents.